(19) RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL
DE LA PROPRIÉTÉ INDUSTRIELLE

PARIS

(11) N° de publication : **2 802 661**
(à n'utiliser que pour les commandes de reproduction)

(21) N° d'enregistrement national : **99 16123**

(51) Int Cl[7] : **G 06 F 7/58**, H 03 K 3/84, G 09 C 1/00

(12) DEMANDE DE BREVET D'INVENTION A1

(22) **Date de dépôt :** 21.12.99.

(30) **Priorité :**

(43) **Date de mise à la disposition du public de la demande :** 22.06.01 Bulletin 01/25.

(56) **Liste des documents cités dans le rapport de recherche préliminaire :** *Se reporter à la fin du présent fascicule*

(60) **Références à d'autres documents nationaux apparentés :**

(71) **Demandeur(s) :** *BULL SA Société anonyme* — FR.

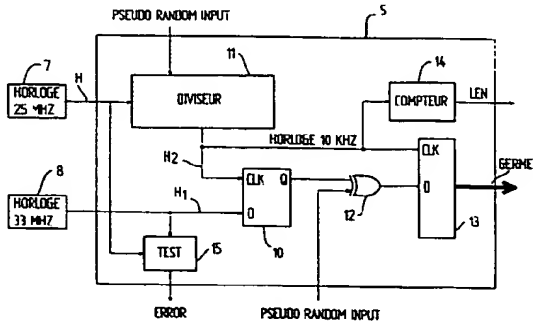
(72) **Inventeur(s) :** LE QUERE PATRICK.

(73) **Titulaire(s) :**

(74) **Mandataire(s) :**

(54) GENERATEUR DE NOMBRES ALEATOIRES HAUT DEBIT.

(57) Générateur physique de nombres aléatoires, caractérisé en ce qu'il comporte un circuit logique (10) comportant au moins une entrée de données (D) et une entrée d'horloge (CLK), l'entrée de données (D) recevant un premier signal d'horloge et l'entrée d'horloge (CLK) recevant un deuxième signal d'horloge différent du premier; et en ce que les deux signaux d'horloge de fréquences différentes proviennent respectivement de deux oscillateurs différents (OSC1 et OC2) travaillant en asynchronisme l'un de l'autre et ne respectant pas le temps d'établissement du circuit logique (10); la sortie du circuit (10) délivrant un signal dans état intermédiaire entre "0 " et " 1 " qualifié de métastable et étant constitué d'une suite de nombres aléatoires.



FR 2 802 661 - A1

Générat ur de n mbres aléatoires haut débit.

La présente invention se situe dans le domaine du chiffrement et concerne plus particulièrement une solution matérielle (hardware) pour l'implémentation d'un générateur de nombres aléatoires destiné notamment à la génération de clés de chiffrement.

Le besoin accru de performance en cryptographie associé à celui d'inviolabilité conduit les fournisseurs de systèmes de sécurité à privilégier des solutions matérielles de plus en plus performante en terme de débit et de qualité d'aléas.

Le générateur selon l'invention, appelé également générateur aléatoire, peut être associé à une carte additionnelle PCI, abréviations anglo-saxonnes pour "Peripheral Component Interconnect" permettant d'accélérer les fonctions cryptographiques d'une machine : serveur ou station.

Une telle carte couplée à un serveur constituera l'élément matériel de sécurité de la machine.

Il existe deux types de générateurs de nombres aléatoires utilisés en électronique.

Le premier type de générateur est basé sur un phénomène physique aléatoire tel que le bruit thermique dans une diode, une émission radioactive, etc. Il est appelé "générateur physique" dans la suite de la description.

Le deuxième type de générateur est basé sur un algorithme alimenté par un "germe", défini ultérieurement, qui fournit en sortie une série de nombres aléatoires avec une période plus ou moins grande. Il est appelé "générateur pseudo-aléatoire" dans la suite de la description.

Une longue période associée à un germe de bonne qualité, en terme de qualité d'aléas, fournit en sortie d'un tel générateur, une série de nombres quasiment imprédictibles.

Les générateurs physiques sont bien sûr les seules sources réelles de nombres aléatoires puisque complètement imprédictibles mais nombre d'entre eux ne sont pas exempt de corrélations au niveau de leur sortie.

D'autre part, leur débit est en général assez faible de l'ordre de quelques dizaines de kilobits par seconde.

Les générateurs pseudo-aléatoires quant à eux sont simples à implémenter sous forme logicielle et permettent de fournir un débit important de nombres aléatoires de l'ordre de quelques dizaines de mégabits par seconde.

Cependant, ce type de générateurs répond à un processus déterministe et est donc prédictible.

La qualité d'un générateur aléatoire est difficile à estimer car il n'existe pas de procédure officielle et normalisée permettant de vérifier le caractère plus ou moins aléatoire d'une suite de nombres.

On trouve cependant deux séries de tests permettant de "valider" un tel générateur.

La première série de tests, appelée tests FIPS140, est décrite dans le document FIPS140-1 intitulé "Security requirements for cryptographic modules" émis par l'organisme américain NIST. Ces tests constituent le minimum exigé pour tout composant de sécurité désireux de revendiquer le label "FIPS140 compliant", un des objectifs de la présente invention.

La deuxième série de tests, mise au point par Georges Marsaglia et appelée tests DIEHARD, sont beaucoup plus exigeants que les tests FIPS et confèrent, au générateur qui les passe tous avec succès, un certain niveau de qualité reconnu.

Ces deux séries de tests sont jointes en annexe de la présente description.

L'invention a notamment pour but de pallier les inconvénients précités et permet de s'affranchir de circuit physique spécifique telle qu'une diode de bruit, tout en répondant au double impératif de débit élevé, supérieur à 100 Mbits/s, et de très bonne qualité des aléas fournis ; qualité mesurée par le fait que le générateur doit passer avec succès les séries de tests FIPS140 et DIEHARD cités ci-dessus.

A cet effet, l'invention a pour premier objet un générateur physique de nombres aléatoires, caractérisé en ce qu'il comporte un circuit logique comportant au moins une entrée de données et une entrée d'horloge, l'entrée de données

recevant un premier signal d'horloge et l'entrée d'horloge recevant un deuxième signal d'horloge différent du premier ; et en ce que les deux signaux d'horloge de fréquences différentes proviennent respectivement de deux oscillateurs différents travaillant en asynchronisme l'un de l'autre et ne respectant pas le temps d'établissement du circuit logique ; la sortie du circuit délivrant un signal dans état intermédiaire entre "0" et "1" qualifié de métastable et étant constitué d'une suite de nombres aléatoires.

L'invention a pour deuxième objet , un générateur de nombres aléatoires haut débit, caractérisé en ce qu'il comporte un générateur physique tel que défini ci-dessus, dont l'entrée de données correspond à l'entrée de données du générateur physique recevant comme signal d'entrée le premier signal "haute fréquence", en ce qu'il comporte un générateur pseudo-aléatoire couplé en sortie du générateur physique recevant sur son entrée un germe délivré par le générateur physique et ré-injectant une partie de son signal de sortie pseudo-aléatoire dans le générateur physique, et en ce qu'il comporte une mémoire interne stockant les nombres aléatoires obtenus en sortie du générateur pseudo-aléatoire ; les deux générateurs fonctionnant à partir d'un même et deuxième signal d'horloge "haute fréquence" généré par un oscillateur externe.

L'invention a pour troisième objet, un mécanisme de génération de nombres aléatoires à la demande, caractérisé en ce qu'il comporte un générateur de nombres aléatoires tel que défini ci-dessus, une mémoire double-port comportant un buffer de réception, couplée en sortie du générateur sur le bus du générateur, et en ce qu'il comporte un microprocesseur, couplé à la mémoire double-port par le bus microprocesseur, communiquant avec le générateur à travers la mémoire double-port et postant dans la mémoire double-port un mot de commande comprenant une adresse et un compte contenant un nombre maximal de mots aléatoires à stocker, et en ce que le buffer de la mémoire double-port, à la demande du microprocesseur, est alimentée par la mémoire interne du générateur jusqu'à épuisement d'un compte correspondant à un nombre maximal déterminé de nombres aléatoires, puis exploité par le microprocesseur.

Enfin, l'invention a pour quatrième objet, une carte accélératrice des fonctions cryptographiques d'une machine informatique, caractérisée en ce qu'elle supporte un générateur de nombres aléatoires ou un mécanisme tels que définis ci-dessus.

L'invention a pour avantage de n'utiliser que des circuits électroniques standard pour la réalisation d'un générateur "physique", et donc de réduire la complexité et le coût d'un tel générateur.

D'autres avantages et caractéristiques de la présente invention apparaîtront à la lecture de la description qui suit faite en référence aux figures annexées qui représentent :

- la figure 1, le principe général d'un mécanisme de génération de nombres aléatoires à la demande, dans lequel s'insère un générateur aléatoire selon l'invention ;
- la figure 2, le schéma bloc d'un générateur aléatoire selon l'invention ;
- la figure 3, une bascule recevant respectivement sur ses entrées, les signaux d'horloge générés par deux oscillateurs de fréquences différentes, et servant à illustrer le phénomène de métastabilité ;
- les figures 4a et 4b, les chronogrammes respectivement des signaux injectés sur les entrées de la bascule de la figure 3 ;
- la figure 4c, le chronogramme du signal de sortie de la bascule ; et
- la figure 5, le schéma bloc d'un générateur physique selon l'invention.

Le principe général d'un mécanisme de génération de nombres aléatoires dans lequel s'insère le générateur selon l'invention est illustré à la figure 1.

Sur cette figure, les liaisons sans flèche sont bidirectionnelles.

Le mécanisme est délimité sur la figure par une ligne fermée discontinue qui peut également délimiter une carte PCI, citée précédemment, supportant le mécanisme.

Le générateur aléatoire 1 est réalisé à partir d'un automate en logique programmable, implémenté dans un composant électronique programmable FPGA, abréviations anglo-saxonnes pour "Field Programmable Gate Array", et

qui sous contrôle d'un microprocesseur 2 délivre à la demande de celui-ci des nombres aléatoires avec un débit élevé (D> 100 Mbits/s).

Ces nombres aléatoires sont utilisés par différents algorithmes et plus particulièrement par les algorithmes de chiffrement pour la génération de clés de chiffrement.

Une mémoire double-port 3, de type DMA, abréviations anglo-saxonnes pour "Direct Memory Acces" est couplée en sortie du générateur aléatoire 1 via le bus du générateur.

Le microprocesseur 2 est couplé à la mémoire double-port 3 via le bus microprocesseur.

Il communique avec le générateur 1 à travers la mémoire double-port 3 qui permet l'échange des données et des commandes/statuts entre le générateur aléatoire 1 et le bus PCI de la machine à laquelle la carte est connectée, via une interface PCI 4.

Le microprocesseur 2 poste dans la mémoire double-port 3 un mot de commande comprenant uniquement une adresse et un compte.

L'adresse pointe vers un buffer de réception 3_1 de la mémoire double-port 3, dans lequel le générateur 1 stocke les mots aléatoires.

Le compte, quant à lui, fixe le nombre de mots aléatoires demandés au générateur 1 avec une capacité maximum, par exemple de 32 Kbytes.

Le microprocesseur 2 envoie alors une commande d'activation de type "chip select" au générateur 1 qui lit le mot de commande dans la mémoire double-port 3 et exécute celui-ci.

Les mots générés sont ensuite stockés dans le buffer de réception 3_1, indiqué par le microprocesseur 2, jusqu'à l'épuisement de la capacité maximum du compte.

L'automate du générateur aléatoire 1 envoie alors une commande d'interruption "interrupt" au microprocesseur 2 lui indiquant que le buffer 3_1 contenant les résultats est disponible en lecture.

Le schéma bloc d'un générateur aléatoire selon l'invention st illustré à la figure 2.

Il comporte principalement deux étages 5 et 6.

Le première étage 5 comporte un générateur "physique" et le deuxième étage 6, couplé au premier, comporte un générateur "pseudo-aléatoire".

Le générateur "physique" est un générateur de nombre aléatoires basé sur les phénomènes physiques de métastabilité et de bruit de phase d'oscillateurs de fréquences différentes.

Le générateur physique 5 fournit au générateur pseudo-aléatoire 6 un germe aléatoire de bonne qualité en ce sens qu'il satisfait aux tests FIPS140 et possède par ailleurs un débit de l'ordre de 10 Kbits/s.

Le générateur "pseudo-aléatoire" 6, à partir du germe reçu du générateur physique 5, implémente un algorithme de type multiplication avec retenue possédant d'une part un très bon débit car implémenté directement en hardware, et satisfaisant d'autre part aux deux séries de test introduits ci-dessus et détaillés en annexe.

Dans le mode de réalisation décrit, les deux générateurs 5 et 6 fonctionnent au rythme d'une horloge externe 7 générant un premier signal d'horloge "haute fréquence" H de fréquence égale à 25 MHz.

Un oscillateur 8 délivre un deuxième signal "haute fréquence" H1 de fréquence égale à 33 MHZ et constitue le signal d'entrée du générateur physique 5.

Une mémoire interne 9 de type FIFO, abréviations anglo-saxonnes pour "First Input First Output", est couplée en sortie du générateur pseudo-aléatoire 6

La mémoire FIFO 9 sert donc à stocker les nombres aléatoires résultant du traitement opéré par les deux générateurs 5 et 6 en attendant que le microprocesseur 2 demande leur transfert vers la mémoire double-port 3.

Le générateur physique selon l'invention exploite le phénomène dit de métastabilité dont le principe est expliqué en détail ci-après en référence aux figures 3 et 4a à 4c.

Lorsque deux oscillateurs OSC1 et OSC2, de fréquence différente, travaillent en asynchronisme l'un de l'autre, ils génèrent chacun un signal d'horloge distinct et les signaux synchronisés à partir de ces signaux d'horloge appartiennent à des domaines d'horloge distincts qui sont en principe

indépendants du point de vue fonctionnel. Il arrive néanmoins un moment où des signaux sont échangés entre les deux domaines. Cette situation est un handicap dans tout design où elle se rencontre car elle conduit au phénomène de métastabilité.

Les signaux issus de domaines d'horloge différents ne respectent pas les temps d'établissement, ou "set-up" en terminologie anglo-saxonne, des éléments mémoires (bascules ou registres) de l'autre domaine conduisant les sorties de ces éléments à prendre des valeurs aléatoires.

On a représenté à la figure 3 une bascule recevant respectivement sur son entrée D et sur son entrée d'horloge, les signaux d'horloge H1 et H2 générés respectivement par les oscillateurs OSC1 et OSC2.

Les figures 4a et 4b illustrent respectivement des exemples de chronogrammes correspondant aux deux signaux d'horloge.

la figure 4c illustre un chronogramme correspondant au signal de sortie Q de la bascule.

La sortie d'une bascule dont le "set-up" a été violé reste dans un état intermédiaire entre l'état "0" et l'état "1" qui est qualifié de métastable avant de se stabiliser dans l'état définitif "0" ou "1".

On règle en général ce problème en mettant deux bascules en série l'une dernière l'autre pour éviter la propagation de cet aléa indésirable.

A l'inverse, cette métastabilité est exploitée par la présente invention et son occurrence est accentuée par l'utilisation de deux oscillateurs de fréquence très différentes.

On choisit ainsi de préférence un signal "haute fréquence" H1 et un signal "basse fréquence" H2.

Le signal "haute fréquence" H1 est échantillonné par le signal " basse fréquence" H2.

Le phénomène recherché est ainsi accentué par deux phénomène physiques :

- le bruit de phase généré par l'oscillateur "haute fréquence"; et
- la variabilité forcée de la période de l'oscillateur "basse fréquence" provoquée par l'utilisation d'une partie des bits d sortie du générateur pseudo-

aléatoire dans le compteur diviseur servant à générer cette fréquence. On obtient ainsi un facteur de forme variable.

La figure 5 illustre le schéma bloc d'un générateur physique 5 selon l'invention dans lequel on retrouve la bascule 10 décrite en référence à la figue 3 pour l'explication de la métastabilité.

Le module d'entrée du générateur physique 5 est un compteur 11 diviseur par 1312.

Il est alimenté par un signal d'horloge externe H, de fréquence égale à 25 MHz, et génère en sortie un signal d'horloge "basse fréquence" H2 qui échantillonne le signal d'horloge "haute fréquence" H1 d'entrée du générateur 5, de fréquence égale à 33 MHz.

Le signal d'horloge H2 est injecté à l'entrée d'horloge CLK de la bascule 10, et le signal H1 est injecté sur l'entrée D de la bascule 10.

Le signal de sortie, obtenu en sortie Q de la bascule 10, est combinée à travers un circuit logique "ou exclusif" 12 avec un bit de sortie du générateur pseudo-aléatoire 6 et envoyé vers l'entrée D d'un registre à décalage 64 bits référencé 13.

Le circuit logique 12 permet ainsi de pallier à une éventuelle défaillance du générateur physique 5.

Le registre à décalage 13 remet en forme le signal issu de la bascule 10.

Dans le mode de réalisation décrit, il génère deux mots aléatoires de 32 bits tous les 32 x 100 μs soit 3,2 ms environ.

Ces deux mots constituent le "germe" utilisé par générateur pseudo-aléatoire 6 décrit ci-après.

Pour répondre notamment aux contraintes de la série de tests DIEHARD, le renouvellement du germe se fait typiquement tous les 100 millions de bits.

Ainsi, un compteur par 3750, 14, génère un signal LEN (Load Enable) qui charge effectivement un nouveau germe dans le générateur pseudo-aléatoire 6 toutes les 375 ms environ.

Le générateur physique 5 comporte en outre un module de test 15 comprenant deux compteurs et un comparateur, non représentés.

Les deux compteurs reçoivent respectivement en entrée les deux signaux d'horloges H et H1, respectivement de 25 MHz et de 33 MHz, et leurs sorties alimentent le comparateur, et génère un signal d'erreur en cas de dysfonctionnement (blocage) de l'horloge à 33 MHz dont le signal est utilisé comme signal externe "haute fréquence" H1.

Ce test permet de vérifier en permanence que la valeur du signal servant à générer le germe aléatoire n'est pas bloqué à "0" ou à "1".

En cas de blocage, le signal d'erreur invalide la sortie finale du générateur physique 5 en forçant l'écriture de zéros dans la mémoire interne 9.

Le générateur pseudo-aléatoire 6 est du type multiplication avec retenue.

Il convient particulièrement à l'implémentation d'un générateur selon l'invention du fait de la rapidité d'exécution de son algorithme mais n'est cependant pas le seul pouvant être utilisé par le générateur pseudo-aléatoire.

L'algorithme s'exprime de la façon suivante :

X := (A * X(15 : 0))+ X(31 : 16) ;

Y := (B * Y(15 : 0))+ Y(31 : 16) ;

PRN(31 : 16) <= X(15 : 0) ;

PRN(15 : 0) <= Y(15 : 0) ;

où :

X et Y sont des variables de 32 bits initialisées avec le germe décrit précédemment, A et B sont des constantes de 16 bits et PRN correspond à un mot de 32 bits délivré en sortie du générateur pseudo-aléatoire 6.

Le mot PRN est envoyé vers la mémoire interne 9 du générateur aléatoire 1 à raison d'un mot de 32 bits toutes les 120 ns soit un débit global de 266 Mbits/s.

La sortie de la mémoire interne 9 couplée à la mémoire double-port 3 alimente le buffer de réception de la mémoire double port 3.

La lecture de la mémoire interne 9 s'effectue à la cadence d'un mot de 32 bits toutes les 30 ns ceci afin de ne pas ralentir le débit interne du générateur aléatoire 1 et ce, même en cas de conflit d'accès à la mémoire double port 3.

En conclusion, la présente invention répond à deux objectifs.

Le premier objectif est de réaliser une implémentation matérielle compacte d'un générateur de nombres aléatoires entièrement réalisé à partir de composants standards n'utilisant pas de composants spécifiques générateur de bruit, et pouvant d'autre part être supportée par une carte PCI accélératrice des ressources cryptographiques d'une machine informatique.

Le deuxième objectif cumulé au premier est de pouvoir répondre aux contraintes de débit et d'aléas recherchées. Dans ce contexte, un générateur "physique" de nombres aléatoires ne se suffit pas à lui-même.

C'est pourquoi la présente invention associe un générateur physique exploitant le phénomène de métastabilité associé à celui du bruit de phase pour garantir un germe de bonne qualité d'aléas, avec un générateur pseudo-aléatoire qui accélère le débit des germes délivrés par le générateur physique et qui, par ailleurs, gomme les éventuelles corrélations en sortie du générateur physique qui n'est pas parfait.

ANNEXES

Ci-joint une description sommaire des tests FIPS140 et DIEHARD passés avec succès par le générateur aléatoire selon l'invention.

1) Les tests FIPS 140 :

Les tests FIPS 140 s'effectuent sur une séquence de 20000 bits et comprennent :

→ un test Monobit : le nombre de bits à "1" doit être tel que : 9654< N <10346

→ un test dit POKER : test qui divise le flux de 20000 bits en 5000 suites contiguës de 4 bits. Pour chaque suite une fonction f(i) est évaluée qui est égale au nombre de fois ou la valeur 0<i<15 est apparue.

La fonction suivante est alors évaluée :

X = (16/5000) * (SOMME (f(i) * f(i))) - 5000 avec 0<i<15

Le test est positif si 1.03 < X <57.4

→ Les Runs tests qui comptabilisent le nombre d'occurrences des suites 11 111 1111 00 000 0000 ...

Le test est positif si pour chaque longueur de run le nombre de résultats est compris dans l'intervalle correspondant

Longueur des runs	Intervalle requis
1	2267-2733
2	1079-1421
3	502-748
4	223-402
5	90-223
>6	90-223

→ Le Long Run Test : qui vérifie l'absence de séquence de 34 "1" ou 34 "0" dans la suite de 20000 bits

Pour plus de détail : voir FIPS PUB 140-1 : SECURITY REQUIREMENTS FOR CRYPTOGRAPHIC MODULE

1) Les tests DIEHARD

Ces tests sont au nombre de 15 et sont tous décrits dans cette annexe . Un fichier contenant 80 Mbits aléatoires est requis pour passer les tests et les résultats sont donnés sous forme de nombres 0<p<1.

Le résultat des tests dépend du nombre de valeurs p= 1 ou p=0 trouvées :

→ Aucune valeur p=1 ou p=0 ➔ le résultat des tests est positif.

→ Quelques valeurs p=1 ou p=0 ➔ le résultat des tests est positif mais l'aléas est de qualité moyenne

→ Le nombre de valeurs p=1 ou p=0 consécutives est > 5 dans un ou plusieurs tests ➔ le résultat des tests est négatif.

```
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::          This is the BIRTHDAY SPACINGS TEST                  ::
:: Choose m birthdays in a year of n days. List the spacings    ::
:: between the birthdays. If j is the number of values that     ::
:: occur more than once in that list, then j is asymptotically  ::
:: Poisson distributed with mean m^3/(4n). Experience shows n   ::
:: must be quite large, say n>=2^18, for comparing the results  ::
:: to the Poisson distribution with that mean. This test uses   ::
:: n=2^24 and m=2^9, so that the underlying distribution for j  ::
:: is taken to be Poisson with lambda=2^27/(2^26)=2. A sample   ::
:: of 500 j's is taken, and a chi-square goodness of fit test   ::
:: provides a p value. The first test uses bits 1-24 (counting  ::
:: from the left) from integers in the specified file.          ::
::  Then the file is closed and reopened. Next, bits 2-25 are   ::
:: used to provide birthdays, then 3-26 and so on to bits 9-32. ::
:: Each set of bits provides a p-value, and the nine p-values   ::
:: provide a sample for a KSTEST.                               ::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
:: THE OVERLAPPING 5-PERMUTATION TEST ::
:: This is the OPERM5 test. It looks at a sequence of one mill- ::
:: ion 32-bit random integers. Each set of five consecutive ::
:: integers can be in one of 120 states, for the 5! possible or- ::
:: derings of five numbers. Thus the 5th, 6th, 7th,...numbers ::
:: each provide a state. As many thousands of state transitions ::
:: are observed, cumulative counts are made of the number of ::
:: occurences of each state. Then the quadratic form in the ::
:: weak inverse of the 120x120 covariance matrix yields a test ::
:: equivalent to the likelihood ratio test that the 120 cell ::
:: counts came from the specified (asymptotically) normal dis- ::
:: tribution with the specified 120x120 covariance matrix (with ::
:: rank 99). This version uses 1,000,000 integers, twice. ::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:: This is the BINARY RANK TEST for 31x31 matrices. The leftmost ::
:: 31 bits of 31 random integers from the test sequence are used ::
:: to form a 31x31 binary matrix over the field {0,1}. The rank ::
:: is determined. That rank can be from 0 to 31, but ranks< 28 ::
:: are rare, and their counts are pooled with those for rank 28. ::
:: Ranks are found for 40,000 such random matrices and a chisqua-::
:: re test is performed on counts for ranks 31,30,29 and <=28. ::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:: This is the BINARY RANK TEST for 32x32 matrices. A random 32x ::
:: 32 binary matrix is formed, each row a 32-bit random integer. ::
:: The rank is determined. That rank can be from 0 to 32, ranks ::
:: less than 29 are rare, and their counts are pooled with those ::
:: for rank 29. Ranks are found for 40,000 such random matrices ::
:: and a chisquare test is performed on counts for ranks 32,31, ::
:: 30 and <=29. ::
```

```
:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:: This is the BINARY RANK TEST for 6x8 matrices. From each of ::
:: six random 32-bit integers from the generator under test, a ::
:: specified byte is chosen, and the resulting six bytes form a ::
:: 6x8 binary matrix whose rank is determined. That rank can be ::
:: from 0 to 6, but ranks 0,1,2,3 are rare; their counts are ::
:: pooled with those for rank 4. Ranks are found for 100,000 ::
:: random matrices, and a chi-square test is performed on ::
:: counts for ranks 6,5 and <=4. ::
:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::                THE BITSTREAM TEST                ::
:: The file under test is viewed as a stream of bits. Call them ::
:: b1,b2,... . Consider an alphabet with two "letters", 0 and 1 ::
:: and think of the stream of bits as a succession of 20-letter ::
:: "words", overlapping. Thus the first word is b1b2...b20, the ::
:: second is b2b3...b21, and so on. The bitstream test counts ::
:: the number of missing 20-letter (20-bit) words in a string of ::
:: 2^21 overlapping 20-letter words. There are 2^20 possible 20 ::
:: letter words. For a truly random string of 2^21+19 bits, the ::
:: number of missing words j should be (very close to) normally ::
:: distributed with mean 141,909 and sigma 428. Thus ::
:: (j-141909)/428 should be a standard normal variate (z score) ::
:: that leads to a uniform [0,1) p value. The test is repeated ::
:: twenty times. ::
:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::          The tests OPSO, OQSO and DNA          ::
::      OPSO means Overlapping-Pairs-Sparse-Occupancy      ::
:: The OPSO test considers 2-letter words from an alphabet of ::
```

```
:: 1024 letters.  Each letter is determined by a specified ten   ::
:: bits from a 32-bit integer in the sequence to be tested. OPSO ::
:: generates  2^21 (overlapping) 2-letter words  (from 2^21+1    ::
:: "keystrokes")  and counts the number of missing words---that  ::
:: is 2-letter words which do not appear in the entire sequence. ::
:: That count should be very close to normally distributed with  ::
:: mean 141,909, sigma 290. Thus (missingwrds-141909)/290 should ::
:: be a standard normal variable. The OPSO test takes 32 bits at ::
:: a time from the test file and uses a designated set of ten    ::
:: consecutive bits. It then restarts the file for the next de-  ::
:: signated 10 bits, and so on.                                  ::
::                                                               ::
::    OQSO means Overlapping-Quadruples-Sparse-Occupancy         ::
::  The test OQSO is similar, except that it considers 4-letter ::
:: words from an alphabet of 32 letters, each letter determined  ::
:: by a designated string of 5 consecutive bits from the test    ::
:: file, elements of which are assumed 32-bit random integers.   ::
:: The mean number of missing words in a sequence of 2^21 four-  ::
:: letter words,  (2^21+3 "keystrokes"), is again 141909, with   ::
:: sigma = 295.  The mean is based on theory; sigma comes from   ::
:: extensive simulation.                                         ::
::                                                               ::
::   The DNA test considers an alphabet of 4 letters:: C,G,A,T,::
:: determined by two designated bits in the sequence of random   ::
:: integers being tested.  It considers 10-letter words, so that ::
:: as in OPSO and OQSO, there are 2^20 possible words, and the   ::
:: mean number of missing words from a string of 2^21  (over-    ::
:: lapping)  10-letter  words (2^21+9 "keystrokes") is 141909.   ::
:: The standard deviation sigma=339 was determined as for OQSO   ::
:: by simulation.  (Sigma for OPSO, 290, is the true value (to   ::
:: three places), not determined by simulation.                  ::
```

```
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::     This is the COUNT-THE-1's TEST on a stream of bytes.      ::
:: Consider the file under test as a stream of bytes (four per   ::
:: 32 bit integer). Each byte can contain from 0 to 8 1's,       ::
:: with probabilities 1,8,28,56,70,56,28,8,1 over 256. Now let   ::
:: the stream of bytes provide a string of overlapping 5-letter  ::
:: words, each "letter" taking values A,B,C,D,E. The letters are ::
:: determined by the number of 1's in a byte:: 0,1,or 2 yield A,::
:: 3 yields B, 4 yields C, 5 yields D and 6,7 or 8 yield E. Thus ::
:: we have a monkey at a typewriter hitting five keys with vari- ::
:: ous probabilities (37,56,70,56,37 over 256). There are 5^5    ::
:: possible 5-letter words, and from a string of 256,000 (over-  ::
:: lapping) 5-letter words, counts are made on the frequencies   ::
:: for each word. The quadratic form in the weak inverse of      ::
:: the covariance matrix of the cell counts provides a chisquare ::
:: test:: Q5-Q4, the difference of the naive Pearson sums of     ::
:: (OBS-EXP)^2/EXP on counts for 5- and 4-letter cell counts.    ::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::     This is the COUNT-THE-1's TEST for specific bytes.        ::
:: Consider the file under test as a stream of 32-bit integers.  ::
:: From each integer, a specific byte is chosen , say the left-  ::
:: most:: bits 1 to 8. Each byte can contain from 0 to 8 1's,    ::
:: with probabilitie 1,8,28,56,70,56,28,8,1 over 256. Now let    ::
:: the specified bytes from successive integers provide a string ::
:: of (overlapping) 5-letter words, each "letter" taking values  ::
:: A,B,C,D,E. The letters are determined by the number of 1's,   ::
:: in that byte:: 0,1,or 2 —> A, 3 —> B, 4 —> C, 5 —> D,::
:: and 6,7 or 8 —> E. Thus we have a monk y at a typewriter      ::
:: hitting five keys with with various probabilities:: 37,56,70,::
```

:: 56,37 over 256. There are 5^5 possible 5-letter words, and ::
:: from a string of 256,000 (overlapping) 5-letter words, counts ::
:: are made on the frequencies for each word. The quadratic form ::
:: in the weak inverse of the covariance matrix of the cell ::
:: counts provides a chisquare test:: Q5-Q4, the difference of ::
:: the naive Pearson sums of (OBS-EXP)^2/EXP on counts for 5- ::
:: and 4-letter cell counts. ::
::
::
:: THIS IS A PARKING LOT TEST ::
:: In a square of side 100, randomly "park" a car—a circle of ::
:: radius 1. Then try to park a 2nd, a 3rd, and so on, each ::
:: time parking "by ear". That is, if an attempt to park a car ::
:: causes a crash with one already parked, try again at a new ::
:: random location. (To avoid path problems, consider parking ::
:: helicopters rather than cars.) Each attempt leads to either ::
:: a crash or a success, the latter followed by an increment to ::
:: the list of cars already parked. If we plot n: the number of ::
:: attempts, versus k:: the number successfully parked, we get a::
:: curve that should be similar to those provided by a perfect ::
:: random number generator. Theory for the behavior of such a ::
:: random curve seems beyond reach, and as graphics displays are ::
:: not available for this battery of tests, a simple characteriz ::
:: ation of the random experiment is used: k, the number of cars ::
:: successfully parked after n=12,000 attempts. Simulation shows ::
:: that k should average 3523 with sigma 21.9 and is very close ::
:: to normally distributed. Thus (k-3523)/21.9 should be a st- ::
:: andard normal variable, which, converted to a uniform varia- ::
:: ble, provides input to a KSTEST based on a sample of 10. ::
::
::

```
::            THE MINIMUM DISTANCE TEST                    ::
:: It does this 100 times:: choose n=8000 random points in a  ::
:: square of side 10000. Find d, the minimum distance between  ::
:: the (n^2-n)/2 pairs of points. If the points are truly inde- ::
:: pendent uniform, then d^2, the square of the minimum distance ::
:: should be (very close to) exponentially distributed with mean ::
:: .995 . Thus 1-exp(-d^2/.995) should be uniform on [0,1) and  ::
:: a KSTEST on the resulting 100 values serves as a test of uni- ::
:: formity for random points in the square. Test numbers=0 mod 5 ::
:: are printed but the KSTEST is based on the full set of 100   ::
:: random choices of 8000 points in the 10000x10000 square.     ::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::            THE 3DSPHERES TEST                           ::
:: Choose  4000 random points in a cube of edge 1000. At each  ::
:: point, center a sphere large enough to reach the next closest ::
:: point. Then the volume of the smallest such sphere is (very  ::
:: close to) exponentially distributed with mean 120pi/3. Thus  ::
:: the radius cubed is exponential with mean 30. (The mean is   ::
:: obtained by extensive simulation). The 3DSPHERES test gener- ::
:: ates 4000 such spheres 20 times. Each min radius cubed leads ::
:: to a uniform variable by means of 1-exp(-r^3/30.), then a    ::
::  KSTEST is done on the 20 p-values.                         ::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::     This is the SQEEZE test                                 ::
::  Random integers are floated to get uniforms on [0,1). Start- ::
::  ing with k=2^31=2147483647, the test finds j, the number of  ::
::  iterations necessary to reduce k to 1, using the reduction   ::
::  k=c  iling(k*U), with U provided by floating integers from   ::
::  the file being tested. Such j's are found 100,000 times,     ::
```

```
:: then counts for the number of times j was <=6,7,...,47,>=48 ::
:: are used to provide a chi-square test for cell frequencies. ::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:: The OVERLAPPING SUMS test ::
:: Integers are floated to get a sequence U(1),U(2),... of uni- ::
:: form [0,1) variables. Then overlapping sums, ::
:: S(1)=U(1)+...+U(100), S2=U(2)+...+U(101),... are formed. ::
:: The S's are virtually normal with a certain covariance mat- ::
:: rix. A linear transformation of the S's converts them to a ::
:: sequence of independent standard normals, which are converted ::
:: to uniform variables for a KSTEST. The p-values from ten ::
:: KSTESTs are given still another KSTEST. ::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:: This is the RUNS test. It counts runs up, and runs down, ::
:: in a sequence of uniform [0,1) variables, obtained by float- ::
:: ing the 32-bit integers in the specified file. This example ::
:: shows how runs are counted: .123,.357,.789,.425,.224,.416,.95::
:: contains an up-run of length 3, a down-run of length 2 and an ::
:: up-run of (at least) 2, depending on the next values. The ::
:: covariance matrices for the runs-up and runs-down are well ::
:: known, leading to chisquare tests for quadratic forms in the ::
:: weak inverses of the covariance matrices. Runs are counted ::
:: for sequences of length 10,000. This is done ten times. Then ::
:: repeated. ::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:: This is the CRAPS TEST. It plays 200,000 games of craps, finds::
:: the number of wins and the number of throws necessary to end ::
:: each game. The numb r of wins should be (very close to) a ::
```

:: normal with mean 200000p and variance 200000p(1-p), with ::
:: p=244/495. Throws necessary to complete the game can vary ::
:: from 1 to infinity, but counts for all>21 are lumped with 21. ::
:: A chi-square test is made on the no.-of-throws cell counts. ::
:: Each 32-bit integer from the test file provides the value for ::
:: the throw of a die, by floating to [0,1), multiplying by 6 ::
:: and taking 1 plus the integer part of the result. ::
::

NOTE: Most of the tests in DIEHARD return a p-value, which should be uniform on [0,1) if the input file contains truly independent random bits. Those p-values are obtained by p=F(X), where F is the assumed distribution of the sample random variable X—often normal. But that assumed F is just an asymptotic approximation, for which the fit will be worst in the tails. Thus you should not be surprised with occasional p-values near 0 or 1, such as .0012 or .9983. When a bit stream really FAILS BIG, you will get p's of 0 or 1 to six or more places. By all means, do not, as a Statistician might, think that a p < .025 or p> .975 means that the RNG has "failed the test at the .05 level". Such p's happen among the hundreds that DIEHARD produces, even with good RNG's. So keep in mind that " p happens".

REVENDICATIONS

1. Générateur physique de nombres aléatoires, caractérisé en ce qu'il comporte un circuit logique (10) comportant au moins une entrée de données (D) et une entrée d'horloge (CLK), l'entrée de données (D) recevant un premier signal d'horloge et l'entrée d'horloge (CLK) recevant un deuxième signal d'horloge différent du premier ; et en ce que les deux signaux d'horloge de fréquences différentes proviennent respectivement de deux oscillateurs différents (OSC1 et OC2) travaillant en asynchronisme l'un de l'autre et ne respectant pas le temps d'établissement du circuit logique (10) ; la sortie du circuit (10) délivrant un signal dans état intermédiaire entre "0" et "1" qualifié de métastable et étant constitué d'une suite de nombres aléatoires.

2. Générateur physique selon la revendication 1, caractérisé en ce le premier signal d'horloge (H1) est un signal "haute fréquence" et en ce que le deuxième signal d'horloge (H2) est un signal "basse fréquence" ; le signal "haute fréquence" étant échantillonné par le signal "basse fréquence" ; la métastabilité du signal obtenu en sortie du circuit (10) étant accentuée par le bruit de phase de l'oscillateur (OSC1) générant le signal "haute fréquence" (H1).

3. Générateur de nombres aléatoires (1) haut débit, caractérisé en ce qu'il comporte un générateur physique (5) selon la revendication 2, dont l'entrée de données correspond à l'entrée de données du générateur physique (5) recevant comme signal d'entrée le signal "haute fréquence" (H1), en ce qu'il comporte un générateur pseudo-aléatoire (6) couplé en sortie du générateur physique (5) recevant sur son entrée un germe délivré par le générateur physique (5) et ré-injectant une partie de son signal de sortie pseudo-aléatoire dans le générateur physique (5), et en ce qu'il comporte une mémoire interne (9) stockant les nombres aléatoires obtenus en sortie du générateur pseudo-aléatoire (6) ; les deux générateurs fonctionnant à partir d'un même et

deuxième signal d'horloge "haute fréquence" généré par un oscillateur extern (7).

4. Générateur de nombres aléatoires selon la revendication 3, caractérisé en ce que le générateur physique (5) comporte un bloc (11) qui partant du signal d'horloge "haute fréquence" , généré par l'oscillateur externe (7), génère le signal "basse fréquence" qui échantillonne le signal d'entrée du générateur physique (5).

5. Générateur de nombres aléatoires selon la revendication 4, caractérisé en ce qu'une partie du signal de sortie du générateur pseudo-aléatoire est ré-injecté dans le bloc (11) générant le signal "basse fréquence" pour forcer la variabilité de la période du signal "basse fréquence".

6. Générateur de nombres aléatoires selon la revendication 5, caractérisé en ce que le générateur physique (5) comporte un registre à décalage (13) recevant sur son entrée d'horloge (CLK) le signal "basse fréquence" généré par le bloc (11) et recevant sur son entrée de données le signal de sortie de la bascule (10) ; le registre à décalage (13) délivrant sur sa sortie le germe alimentant le générateur pseudo-aléatoire (6).

7. Générateur de nombres aléatoires selon la revendication 6, caractérisé en ce que le générateur physique (5) comporte en outre une porte logique "ou exclusif" (12), couplée entre la bascule (10) et le registre à décalage (13), et recevant sur une première entrée le signal de sortie de la bascule (10) et sur une deuxième entrée, un bit du signal de sortie du générateur aléatoire (6) pour pallier à une éventuelle défaillance du générateur physique (5).

8. Générateur de nombres aléatoires selon la revendication 7, caractérisé n ce que le générateur physique (5) comporte un compteur (14) recevant sur son entrée le signal d'horloge "basse fréquence" et dont le signal de sortie

commande la cadence du renouvellement du germe dans le générateur pseudo-aléatoire (6).

9. Générateur de nombres aléatoires selon l'une quelconque des revendications 3 à 8, caractérisé en ce qu'il comporte en outre, un module de test (15) couplé en entrée du générateur physique (5) qui reçoit en entrée les signaux "basse fréquence" et "haute fréquence" et délivre en sortie un signal d'erreur en cas de dysfonctionnement de l'oscillateur délivrant le signal "haute fréquence" qui invalide la sortie du générateur physique en forçant l'écriture de zéros dans la mémoire interne (9).

10. Générateur de nombres aléatoires selon l'une quelconque des revendications 3 à 9, caractérisé en ce que le générateur pseudo-aléatoire (6) implémente un algorithme du type multiplication avec retenue.

11. Générateur de nombres aléatoires selon l'une quelconque des revendications 4 à 10, caractérisé en ce qu'il est entièrement réalisé à partir d'un composant FPGA.

12. Mécanisme de génération de nombres aléatoires à la demande, caractérisé en ce qu'il comporte un générateur de nombres aléatoires (1) selon l'une quelconque des revendications 3 à 11, une mémoire double-port (3) comportant un buffer de réception (3_1), couplée en sortie du générateur (1) sur le bus du générateur (1), et en ce qu'il comporte un microprocesseur (2), couplé à la mémoire double-port (3) par le bus microprocesseur, communiquant avec le générateur (1) à travers la mémoire double-port (3) et postant dans la mémoire double-port (3) un mot de commande comprenant une adresse et un compte contenant un nombre maximal de mots aléatoires à stocker, et en ce que l buffer (3_1) de la mémoire double-port (3), à la demande du microprocesseur (2), est alimentée par la mémoire interne (9) du générateur

(1) jusqu'à épuisement d'un compte correspondant à un nombre maximal déterminé de nombres aléatoires, puis exploité par le microprocesseur (2).

13. Carte accélératrice des fonctions cryptographiques d'une machine informatique, caractérisée en ce qu'elle supporte un générateur aléatoire selon l'une quelconque des revendications 3 à 11.

14. Carte accélératrice des fonctions cryptographiques d'une machine informatique, caractérisée en ce qu'elle supporte un mécanisme selon la revendications 12.

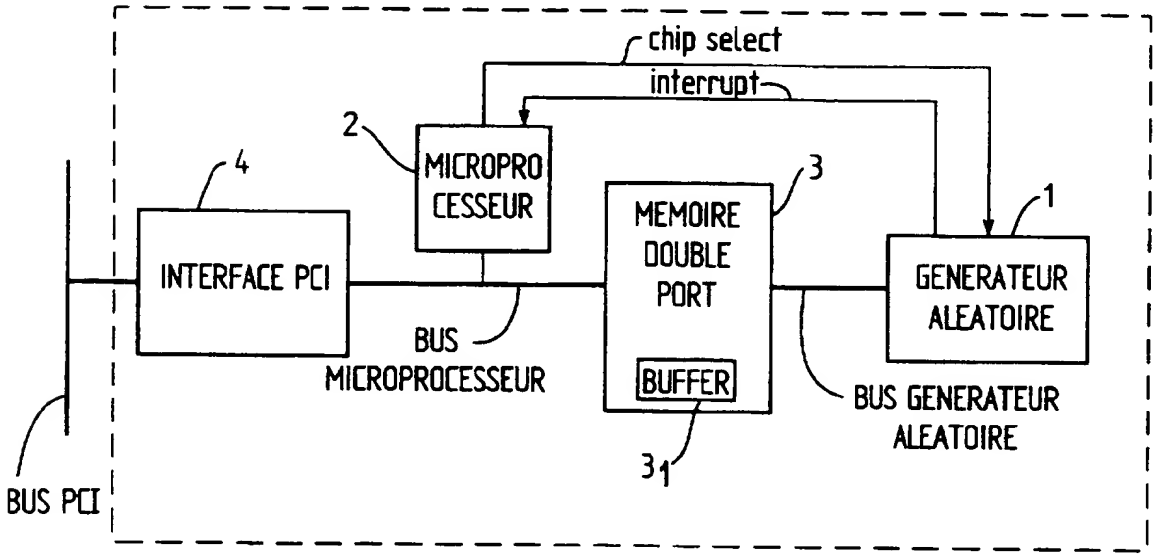


FIG.1

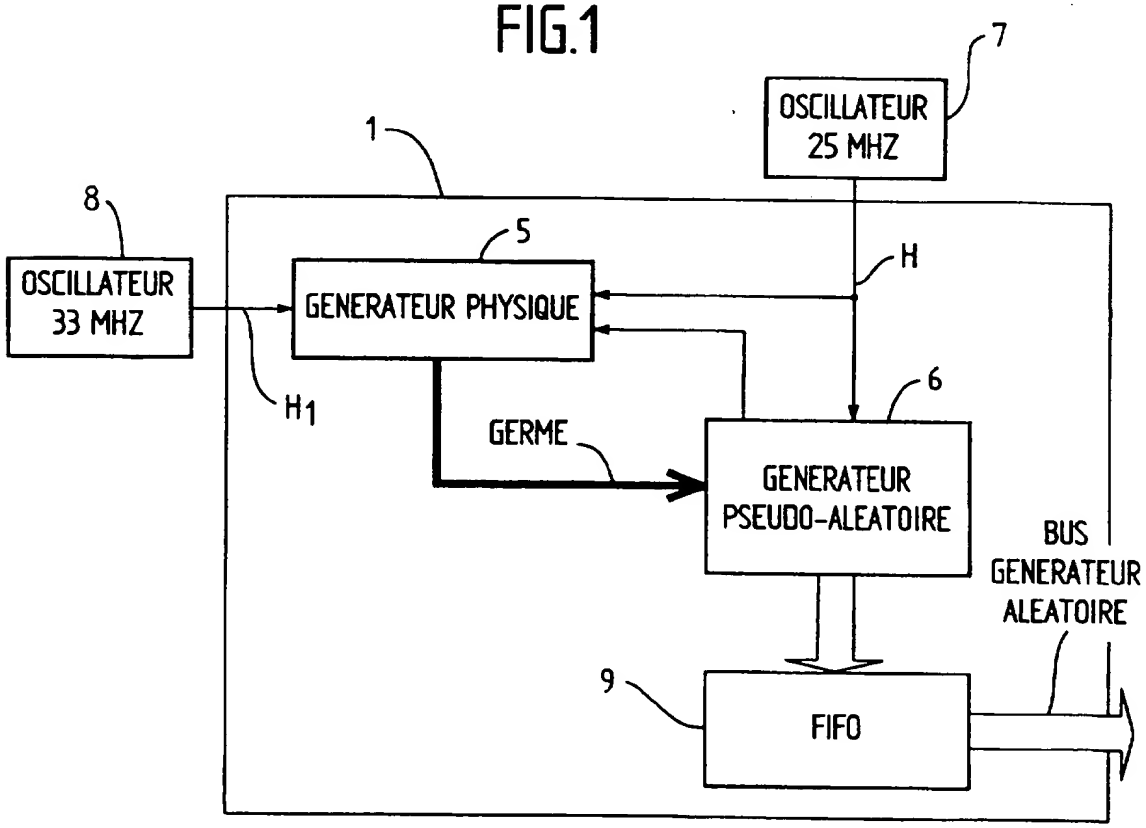


FIG.2

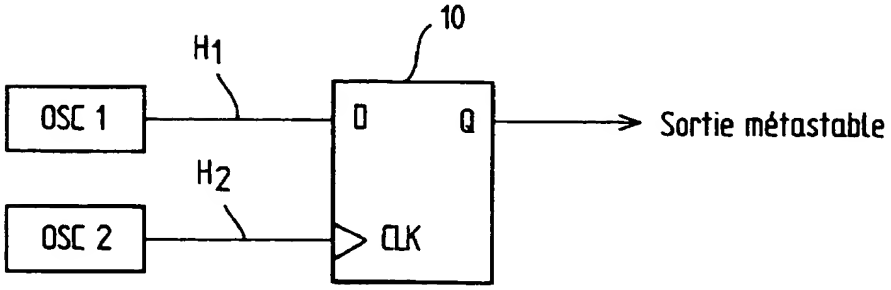
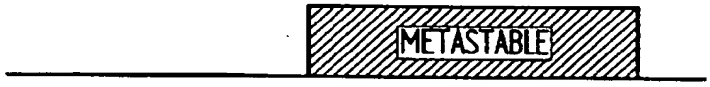


FIG.3

FIG.4a

FIG.4b

FIG.4c



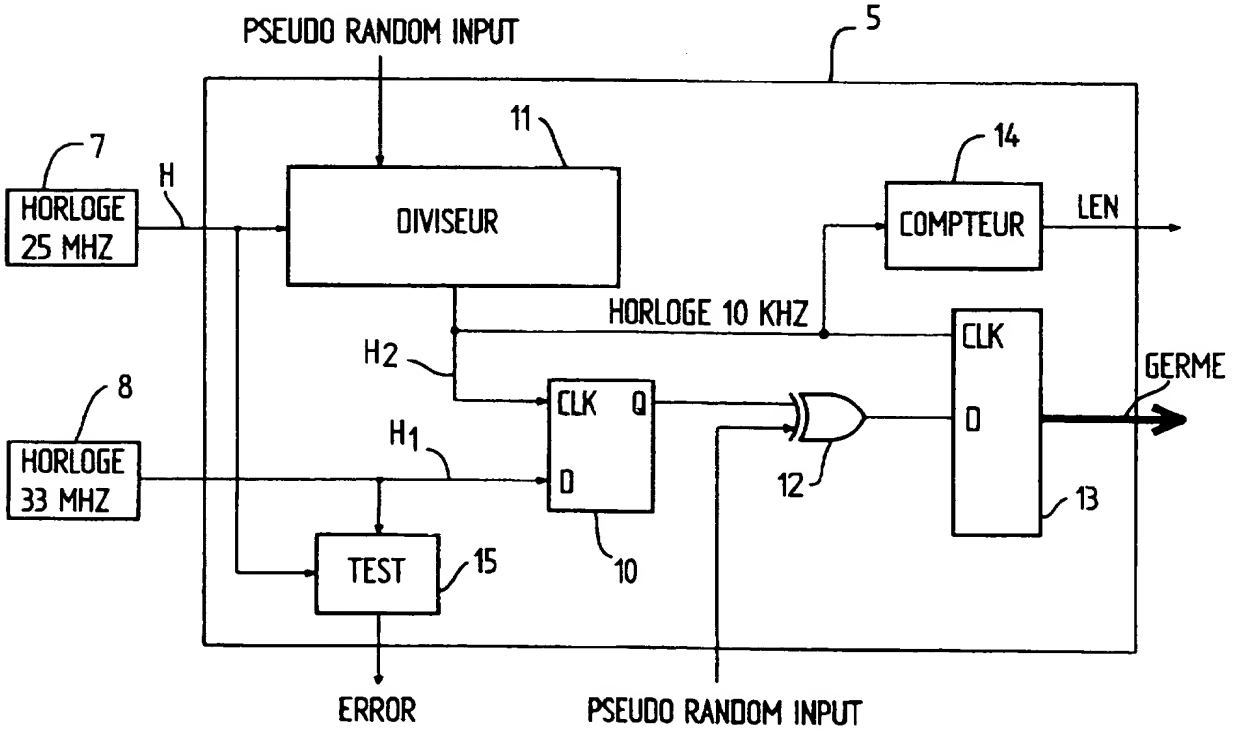


FIG.5

RÉPUBLIQUE FRANÇAISE

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

RAPPORT DE RECHERCHE PRÉLIMINAIRE

établi sur la base des dernières revendications déposées avant le commencement de la recherche

2802661

N° d'enregistrement national

FA 583200
FR 9916123

DOCUMENTS CONSIDÉRÉS COMME PERTINENTS		Revendication(s) concernée(s)	Classement attribué à l'invention par l'INPI
Catégorie	Citation du document avec indication, en cas de besoin, des parties pertinentes		
X	WO 99 61978 A (ABLE TECHNOLOGIES INC N) 2 décembre 1999 (1999-12-02) * le document en entier *	1-3	G06F7/58 H03K3/84 G09C1/00
X	US 4 641 102 A (COULTHART KENNETH B ET AL) 3 février 1987 (1987-02-03)	1,2	
A	* abrégé * * colonne 2, ligne 53 - colonne 3, ligne 3 * * colonne 4, ligne 36 - ligne 49 *	3	
X	"INTEGRATED CIRCUIT COMPATIBLE RANDOM NUMBER GENERATOR" IBM TECHNICAL DISCLOSURE BULLETIN,US,IBM CORP. NEW YORK, vol. 30, no. 11, 1 avril 1988 (1988-04-01), pages 333-335, XP000021682 ISSN: 0018-8689	1,2	
A	* le document en entier *	4	
X	PETRIE C S ET AL: "MODELING AND SIMULATION OF OSCILLATOR-BASED RANDOM NUMBER GENERATORS" IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS),US,NEW YORK, IEEE, 12 mai 1996 (1996-05-12), pages 324-327, XP000704602 ISBN: 0-7803-3074-9 * page 324 *	1,2	

DOMAINES TECHNIQUES RECHERCHÉS (Int.CL.7)

G06F
H03K

1

Date d'achèvement de la recherche	Examinateur
31 août 2000	Verhoof, P

CATÉGORIE DES DOCUMENTS CITÉS

X : particulièrement pertinent à lui seul
Y : particulièrement pertinent en combinaison avec un autre document de la même catégorie
A : arrière-plan technologique
O : divulgation non-écrite
P : document intercalaire

T : théorie ou principe à la base de l'invention
E : document de brevet bénéficiant d'une date antérieure à la date de dépôt et qui n'a été publié qu'à cette date de dépôt ou qu'à une date postérieure.
D : cité dans la demande
L : cité pour d'autres raisons

& : membre de la même famille, document correspondant

EPO FORM 1503 12.99 (P04C14)